LAPORAN PRAKTIKUM DATA STRUCTURE

SIMPLE SORTING

Dosen Pengampu:

H. Fatchurrochman,M.Kom

Asisten Praktikum:

Fillah Anjany 230605110033

Oleh :

Muhammad Alif Mujaddid **240605110082**

Jurusan Teknik Informatika Fakultas Sains dan Teknologi

UIN Maulana Malik Ibrahim Malang

2025

A. Pendahuluan

1. Algoritma sorting yang paling sederhana adalah bubble sort. Langkah sorting secara ascending menggunakan bubble sort adalah sebagai berikut:

a. Bandingkan dua item

b. Jika item pertama (sebelah kiri) lebih besar daripada item kedua (sebelah kanan), maka tukar kedua item tersebut

c. Pindah ke posisi kanan. Lakukan langkah a dan b hingga posisi sampai di batas akhir

d. Ketika satu item telah berada sesuai urutan, ulangi langkah a-c hingga semua item sesuai urutan.

```
...
public void BubbleSort() {
int batas, i;
for (batas = nElemen-1; batas>0; batas--) {
    for (i = 0; i < batas; i++) {
        if (arr[i] > arr[i + 1]) {
            swap(i, i + 1);
        }
    }
}
}

public void swap(int one, int two) {
    int temp = arr[one];
    arr[one] = arr[two];
    arr[two] = temp;
}
...
```

Deklarasi Methode

Membuat variable batas dan iterasi

Membuat itertasi dari Panjang data sampai 0 (dec)

Inner loop dari 0 sampai batas

Jika arr ke I lebih besar dari setelahnya :

Tukar array I dengan setelahnya

Deklarasi methode untuk menukar data 1 dg sebelahnya

Buat variable sementara untuk menimpa variable yang akan diturkar

Masukan arr[two] ke arr[one]

Masukan temp ke arr[two]

Listing diatas adalah baris kode yang mengimplementasikan tiap langkah bubble sort. Implementasikan baris kode tersebut ke dalam sebuah program sehingga dapat digunakan untuk mengurutkan elemen array. Anda dapat menambahkan listing

tersebut pada class HighArray (listing 5 praktikum 1) yang dipanggil pada class HighArrayApp. Atau anda juga dapat menuliskan listing ini pada satu class secara terstruktur. Insert 6 item pada array program tersebut. Tampilkan isi array sebelum dilakukan pengurutan. Kemudian urutkan dan tampilkan isi array setelah pengurutan. Jalankan program yang telah Anda buat dan tuliskan output program tersebut.

```
54      public void BubbleSort(){
55          int batas, i;
56          for ( batas = nElemen-1; batas >0; batas--) {
57              for ( i = 0; i < nElemen; i++) {
58                  if(arr[i] > arr[i+1]){
59                  swap(i, i+1);
60                  }
61              }
62          }
63      }
64      public void swap(int one,int two){
65          int temp = arr[one];
66          arr[one] = arr[two];
67          arr[two] = temp;
68      }
69  }
```

```java
public class HighArrayApp {
    Run | Debug
    public static void main(String[] args) {
    int maxSize = 100;
    HighArray arr;
    arr = new HighArray(maxSize);
    arr.insert(value:70);
    arr.insert(value:25);
    arr.insert(value:75);
    arr.insert(value:80);
    arr.insert(value:55);
    arr.insert(value:30);
    arr.insert(value:0);
    arr.insert(value:85);
    arr.insert(value:40) ;
    arr.insert(value:90);
    arr.insert(value:1);
    arr.insert(value:100);
    arr.display();
    System.out.println(x:"sesudah diurutkan : ");
    arr.BubbleSort();
    arr.display();
```

Output :

```
addid@LAPTOP-F80MDN28 MINGW64 /a/Code/DSA (main)
$  cd a:\\Code\\DSA ; /usr/bin/env C:\\Program\ Files\\Java\\jdk-22\\bin\\java.exe -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:57940 -XX:+ShowCodeDetailsInExceptionMessages -cp C:\\Users\\addid\\AppData\\Roaming\\Code\\User\\workspaceStorage\\df75c84a286eacae2dce0b54ebb13767\\redhat.java\\jdt_ws\\DSA_2f521fc6\\bin Praktikum.HighArrayApp
70 25 75 80 55 30 0 85 40 90 1 100
sesudah diurutkan :
0 0 1 25 30 40 55 70 75 80 85 90
```

2. Untuk sorting array yang memiliki 6 item, berapa jumlah perbandingan item yang dilakukan hingga semua item sesuai urutan?

Jumlah Perbandingan=n(n−1) / 2

Jumlah Perbandingan=6×(6−1)/2=6×5/2=15

3. Pada program nomer 1, tambahkan kode untuk menampilkan isi array setelah baris kode pertukaran item (swap(i, i + 1);). Jalankan kembali dan amati bagaimana proses pengurutan pada tiap iterasi. Tuliskan isi array pada 10 perulangan pertama. Jelaskan!

```
public void BubbleSort(){
    int batas, i;
    for ( batas = nElemen-1; batas >0; batas--) {
        for ( i = 0; i < nElemen; i++) {
            if(arr[i] > arr[i+1]){
            swap(i, i+1);
            display();
            }
        }
    }
public void swap(int one,int two){
    int temp = arr[one];
    arr[one] = arr[two];
    arr[two] = temp;
}
}
```

```
70 25 75 80 55 30 0 85 40 90 1 100
sesudah diurutkan :
25 70 75 80 55 30 0 85 40 90 1 100
25 70 75 55 80 30 0 85 40 90 1 100
25 70 75 55 30 80 0 85 40 90 1 100
25 70 75 55 30 0 80 85 40 90 1 100
25 70 75 55 30 0 80 40 85 90 1 100
25 70 75 55 30 0 80 40 85 1 90 100
25 70 75 55 30 0 80 40 85 1 90 0
25 70 55 75 30 0 80 40 85 1 90 0
25 70 55 30 75 0 80 40 85 1 90 0
25 70 55 30 0 75 80 40 85 1 90 0
25 70 55 30 0 75 40 80 85 1 90 0
25 70 55 30 0 75 40 80 1 85 90 0
25 70 55 30 0 75 40 80 1 85 0 90
25 55 70 30 0 75 40 80 1 85 0 90
25 55 30 70 0 75 40 80 1 85 0 90
25 55 30 0 70 75 40 80 1 85 0 90
25 55 30 0 70 40 75 80 1 85 0 90
25 55 30 0 70 40 75 1 80 85 0 90
25 55 30 0 70 40 75 1 80 0 85 90
25 30 55 0 70 40 75 1 80 0 85 90
25 30 0 55 70 40 75 1 80 0 85 90
25 30 0 55 40 70 75 1 80 0 85 90
25 30 0 55 40 70 1 75 80 0 85 90
25 30 0 55 40 70 1 75 0 80 85 90
25 0 30 55 40 70 1 75 0 80 85 90
25 0 30 40 55 70 1 75 0 80 85 90
25 0 30 40 55 1 70 75 0 80 85 90
25 0 30 40 55 1 70 0 75 80 85 90
0 25 30 40 55 1 70 0 75 80 85 90
0 25 30 40 1 55 70 0 75 80 85 90
0 25 30 40 1 55 0 70 75 80 85 90
0 25 30 1 40 55 0 70 75 80 85 90
0 25 30 1 40 0 55 70 75 80 85 90
0 25 1 30 40 0 55 70 75 80 85 90
0 25 1 30 0 40 55 70 75 80 85 90
0 1 25 30 0 40 55 70 75 80 85 90
0 1 25 0 30 40 55 70 75 80 85 90
0 1 0 25 30 40 55 70 75 80 85 90
0 0 1 25 30 40 55 70 75 80 85 90
0 0 1 25 30 40 55 70 75 80 85 90
```

Bubble Sort itu kayak gelembung di air. Setiap kali gelembung bergerak, dia bakal naik ke atas pelan-pelan. Nah, di sini yang naik ke atas itu angka yang paling besar.

Caranya:

1. Bandingkan dua angka yang bersebelahan.
2. Kalau yang kiri lebih besar dari yang kanan => tukar tempat.
3. Ulangi terus sampai selesai 1 putaran => angka paling besar sudah ada di ujung kanan.
4. Lanjut lagi putaran berikutnya, sekarang angka terbesar kedua naik ke sampingnya.
5. Begitu terus sampai semua angka rapi.

4. Jika Anda ingin melakukan sorting menggunakan algoritma bubble sort secara descending, maka pada program nomer 1 bagian manakah yang harus diganti. Tuliskan code-nya dan jelaskan!

```
public void BubbleSort(){
    int batas, i;
    for ( batas = nElemen-1; batas >0; batas--) {
        for ( i = 0; i < nElemen; i++) {
            if(arr[i] < arr[i+1]){
            swap(i, i+1);
            // display();
            }
        }
    }
}
public void swap(int one,int two){
    int temp = arr[one];
    arr[one] = arr[two];
    arr[two] = temp;
}
}
```

Output :

```
70 25 75 80 55 30 0 85 40 90 1 100
sesudah diurutkan :
100 90 85 80 75 70 55 40 30 25 1 0
```

Jadi yang diganti itu dipengecekanya jadi kurang dari, karena jika kita lihat semisal 70 < 80 maka true dan akan dilakukan swap yang mana 80 akan berada di kanan.

5. Algoritma sorting yang lain adalah selection sort. Langkah selection sort untuk pengurutan secara ascending yaitu: a. Cari item terkecil pada array b. Letakkan item terkecil sesuai urutannya dengan cara menukar item terkecil dengan item pada index awal c. Geser posisi awal pencarian ke kanan. d. Lakukan langkah a, b, dan c hingga semua item terurut. Berikut ini listing untuk algoritma selection sort. Lengkapi sebagaimana soal nomor 1, gunakan data array yang sama, jalankan dan jelaskan tiap baris code pengurutan berikut ini!

```
...
public void SelectionSort() {
    int awal, i, min;

    for (awal=0; awal< nElemen-1; awal++) {
        min = awal;
        for (i = awal + 1; i < nElemen; i++) {
            if (arr[i] < arr[min]) {
                min = i;
            }
        }
        swap(awal, min);
    }
}
...
```

Deklarasi variabel:awal = posisi elemen yang lagi dicek.i = buat jalan ngecek elemen lain. min = index nilai terkecil yang ketemu.

Mulai dari elemen pertama sampai elemen terakhir. Setiap langkah, kita anggap posisi awal itu yang mau ditaruh nilai terkecil.

Anggap dulu nilai terkecil ada di posisi awal.

Bandingkan elemen setelah awal dengan elemen terkecil sementara (min). Kalau nemu elemen yang lebih kecil, ganti min jadi posisi elemen itu. Jadi sekarang min nunjuk ke elemen terkecil baru.

Tukar elemen di posisi awal dengan elemen terkecil yang ketemu (min). Jadi elemen terkecil pindah ke depan.

```
    public void selectionSort() {
    int awal, i, min;
    for (awal=0; awal< nElemen-1; awal++) {
        min = awal;
        for (i = awal + 1; i < nElemen; i++) {
            if (arr[i] < arr[min]) {
            min = i;
        }
    }
        swap(awal, min);
    }
```

6. Pada listing nomor 5, tambahkan kode untuk menampilkan isi elemen pada array setelah kode pertukaran (swap(awal, min);). Jalankan program, amati dan tulis outputnya, kemudian jelaskan!

```
public void selectionSort() {
    int awal, i, min;
    for (awal = 0; awal < nElemen - 1; awal++) {
        min = awal;
        for (i = awal + 1; i < nElemen; i++) {
            if (arr[i] < arr[min]) {
                min = i;
            }
        }
        swap(awal, min);
        display();
    }
}

}
```

```
70 25 75 80 55 30 0 85 40 90 1 100
sesudah diurutkan :
0 25 75 80 55 30 70 85 40 90 1 100
0 1 75 80 55 30 70 85 40 90 25 100
0 1 25 80 55 30 70 85 40 90 75 100
0 1 25 30 55 80 70 85 40 90 75 100
0 1 25 30 40 80 70 85 55 90 75 100
0 1 25 30 40 55 70 85 80 90 75 100
0 1 25 30 40 55 70 85 80 90 75 100
0 1 25 30 40 55 70 75 80 90 85 100
0 1 25 30 40 55 70 75 80 90 85 100
0 1 25 30 40 55 70 75 80 85 90 100
0 1 25 30 40 55 70 75 80 85 90 100
0 1 25 30 40 55 70 75 80 85 90 100
```

Jadi penjelasan gampangnya tuh si array yang paling kiri dibandingkan paling kanan, kalau ketemu angka yang lebih kecil, langsung ditandain dulu, terus di akhir tuker posisinya. Habis itu geser ke elemen berikutnya di kiri, terus ulang lagi sampai semua elemen diurutkan.

7. Sedikit berbeda dengan dua algoritma sebelumnya, pada insertion sort, aksi yang dilakukan adalah membandingkan dan meng-copy item. Berikut ini langkah insertion sort untuk pengurutan secara ascending:

a. Tandai sebuah item sebagai batas antara partially sorted dan unsorted items.

b. Geser item pada partially sorted yang bernilai lebih besar dari pada item yang ditandai pada langkah a.

c. Sisipkan item tersebut pada posisi yang sesuai di bagian partially sorted.

d. Ulangi langkah a-c hingga semua unsorted items telah disisipkan (insert) ke sorted group. Berikut ini listing untuk algoritma insertion sort.

Lengkapi sebagaimana soal nomor 1, gunakan data array yang sama, jalankan dan jelaskan tiap baris code pengurutan berikut ini!

```
...
public void InsertionSort() {
  int i, curIn;

  for (curIn= 1; curIn < nElemen; curIn++) {
      int temp = arr[curIn];
```

Deklarasi Method insert

Deklarasi variable I untuk geser dan curin untuk yg dicek

Ambil angka baru: mulai dari elemen ke-1 (karena elemen pertama dianggap sudah “rapi”), simpan di temp buat dibandingin.

```
        i = curIn;
        while (i > 0 && arr[i - 1] >= temp) {
            arr[i] = arr[i - 1];
            i--;
        }
        arr[i] = temp;
    }
}
...
```

Geser elemen ke kanan: selama angka di kiri (arr[i-1]) lebih besar dari angka yang lagi dipegang (temp), geser elemen itu ke kanan, terus mundur satu langkah.

Taruh angka di posisi yang pas: setelah ketemu tempat yang tepat, masukkan angka yang tadi dipegang (temp).

```
public void InsertionSort() {
    int i, curIn;
    for (curIn = 1; curIn < nElemen; curIn++) {
        int temp = arr[curIn];
        i = curIn;
        while (i > 0 && arr[i - 1] > temp) {
            arr[i] = arr[i - 1];
            i--;
        }
        arr[i] = temp;
    }
}
```

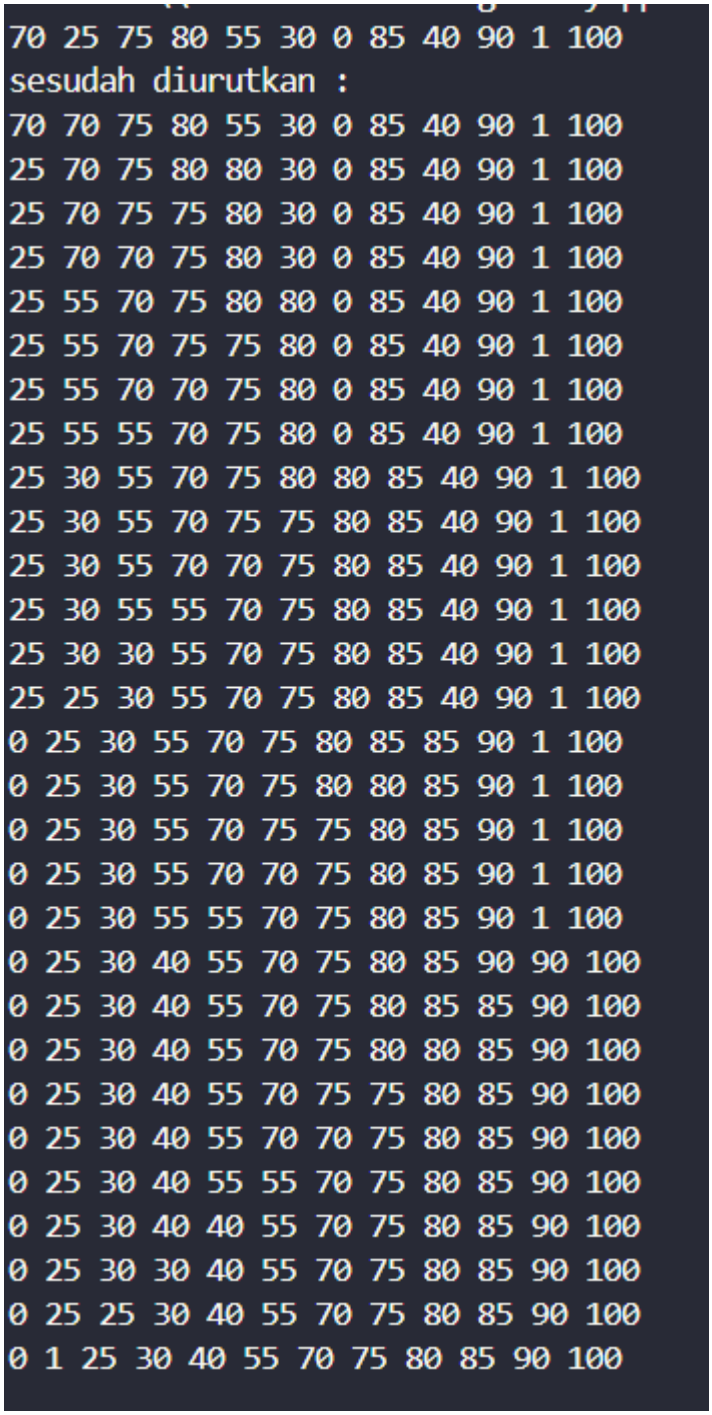
Output :

```
70 25 75 80 55 30 0 85 40 90 1 100
sesudah diurutkan :
0 1 25 30 40 55 70 75 80 85 90 100
```

8. Pada listing nomor 7, tambahkan kode untuk menampilkan isi elemen pada array setelah kode pergeseran item (arr[i] = arr[i - 1];) dan setelah kode insert item (arr[i] = temp;). Jalankan program, amati dan tulis outputnya hingga 4 kali tahap penyisipan, kemudian jelaskan!

```java
public void insertionSort() {
    int i, curIn;
    for (curIn = 1; curIn < nElemen; curIn++) {
        int temp = arr[curIn];
        i = curIn;
        while (i > 0 && arr[i - 1] > temp) {
            arr[i] = arr[i - 1];
            display();
            i--;
        }
        arr[i] = temp;
    }
}

}
```



Jadi penjelasan gampangnya tuh si array jalan dari kiri ke kanan. Setiap ambil angka baru, dia dibandingin sama angka-angka di sebelah kirinya. Kalau ada yang lebih besar, angka itu digeser ke kanan terus sampai ketemu posisi yang pas. Baru deh angka yang dipegang tadi ditaruh di situ. Ulang terus sampai semua angka rapi dari kecil ke besar.